1. For the first step upload target file first and visualize frist 5 rows using head()

```python
if os.environ.get('RUNTIME_ENV_LOCATION_TYPE') == 'external':
    endpoint_517c4add7ff840f8915f46a7a0f89295 = 'https://s3.ap.cloud-object-storage.appdomain.cloud'
else:
    endpoint_517c4add7ff840f8915f46a7a0f89295 = 'https://s3.private.ap.cloud-object-storage.appdomain.cloud'

client_517c4add7ff840f8915f46a7a0f89295 = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='jrUiIrk68Cc8AeuNY0kkYbwC-HP2b_RQ2y905E3I6Wvu',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url=endpoint_517c4add7ff840f8915f46a7a0f89295)

body = client_517c4add7ff840f8915f46a7a0f89295.get_object(Bucket='pythonprojectfordatascience-donotdelete-pr-i4fyreja3rirtd',Key='Sacramentorealestatetra
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

df_data_2 = pd.read_csv(body)
df_data_2.head()
```

Out[1]:

	street	city	zip	state	beds	baths	sq__ft	type	sale_date	price	latitude	longitude
0	3526 HIGH ST	SACRAMENTO	95838	CA	2	1	836	Residential	Wed May 21 00:00:00 EDT 2008	59222	38.631913	-121.434879
1	51 OMAHA CT	SACRAMENTO	95823	CA	3	1	1167	Residential	Wed May 21 00:00:00 EDT 2008	68212	38.478902	-121.431028
2	2796 BRANCH ST	SACRAMENTO	95815	CA	2	1	796	Residential	Wed May 21 00:00:00 EDT 2008	68880	38.618305	-121.443839
3	2805 JANETTE WAY	SACRAMENTO	95815	CA	2	1	852	Residential	Wed May 21 00:00:00 EDT 2008	69307	38.616835	-121.439146
4	6001 MCMAHON DR	SACRAMENTO	95824	CA	2	1	797	Residential	Wed May 21 00:00:00 EDT 2008	81900	38.519470	-121.435768

2. Taking some info from table

In [2]:
```python
df_data_2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 985 entries, 0 to 984
Data columns (total 12 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   street     985 non-null    object 
 1   city       985 non-null    object 
 2   zip        985 non-null    int64  
 3   state      985 non-null    object 
 4   beds       985 non-null    int64  
 5   baths      985 non-null    int64  
 6   sq__ft     985 non-null    int64  
 7   type       985 non-null    object 
 8   sale_date  985 non-null    object 
 9   price      985 non-null    int64  
 10  latitude   985 non-null    float64
 11  longitude  985 non-null    float64
dtypes: float64(2), int64(5), object(5)
memory usage: 92.5+ KB
```

As we can see all rows doesn't have null value, so we can prepare the data ASAP

3. Seeing Description from table

```
In [3]: df_data_2.describe()
```

Out[3]:

	zip	beds	baths	sq__ft	price	latitude	longitude
count	985.000000	985.000000	985.000000	985.000000	985.000000	985.000000	985.000000
mean	95750.697462	2.911675	1.776650	1314.916751	234144.263959	38.607732	-121.355982
std	85.176072	1.307932	0.895371	853.048243	138365.839085	0.145433	0.138278
min	95603.000000	0.000000	0.000000	0.000000	1551.000000	38.241514	-121.551704
25%	95660.000000	2.000000	1.000000	952.000000	145000.000000	38.482717	-121.446127
50%	95762.000000	3.000000	2.000000	1304.000000	213750.000000	38.626582	-121.376220
75%	95828.000000	4.000000	2.000000	1718.000000	300000.000000	38.695589	-121.295778
max	95864.000000	8.000000	5.000000	5822.000000	884790.000000	39.020808	-120.597599

4. Import necessary library

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
```

5. Change type column and city column into integer using labelencoder and knowing their index number as well using dataframe.

```
# we are going to convert 'type' & 'city' columns into number
type_n = pd.DataFrame(set(df_data_2['type']), columns = ['type'])
city_n = pd.DataFrame(set(df_data_2['city']), columns = ['cities number'])
print(type_n)
print(city_n)
```

```
          type
0  Residential
1        Condo
2       Unkown
3 Multi-Family
```

```
        cities number
0         CAMERON PARK
1              ROCKLIN
2       RANCHO MURIETA
3       CITRUS HEIGHTS
4               AUBURN
5           GOLD RIVER
6      WEST SACRAMENTO
7        POLLOCK PINES
8          SLOUGHHOUSE
9                 GALT
10           ROSEVILLE
11              MATHER
12        WALNUT GROVE
13     DIAMOND SPRINGS
14          GREENWOOD
15          RIO LINDA
16             ELVERTA
17     NORTH HIGHLANDS
18     SHINGLE SPRINGS
19          SACRAMENTO
20             LINCOLN
21              FOLSOM
22      RANCHO CORDOVA
23          FORESTHILL
24           EL DORADO
25              WILTON
26              LOOMIS
27     EL DORADO HILLS
28                COOL
29          CARMICHAEL
30         PLACERVILLE
31        MEADOW VISTA
32              PENRYN
33           ANTELOPE
```

In [6]:
```python
labelencoder = LabelEncoder()
```

In [7]:
```python
df_data_2['type'] = labelencoder.fit_transform(df_data_2['type'])
df_data_2['city'] = labelencoder.fit_transform(df_data_2['city'])
```

In [8]:
```python
print(set(df_data_2['type']))
print(set(df_data_2['city']))
```

And the result is

]:
```python
df_data_2
```

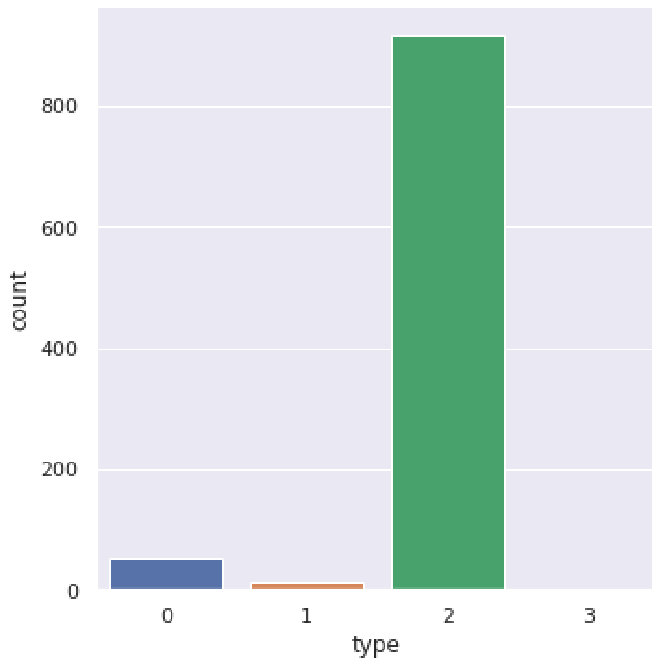
[9]:

	street	city	zip	state	beds	baths	sq__ft	type	sale_date	price	latitude	longitude
0	3526 HIGH ST	33	95838	CA	2	1	836	2	Wed May 21 00:00:00 EDT 2008	59222	38.631913	-121.434879
1	51 OMAHA CT	33	95823	CA	3	1	1167	2	Wed May 21 00:00:00 EDT 2008	68212	38.478902	-121.431028
2	2796 BRANCH ST	33	95815	CA	2	1	796	2	Wed May 21 00:00:00 EDT 2008	68880	38.618305	-121.443839
3	2805 JANETTE WAY	33	95815	CA	2	1	852	2	Wed May 21 00:00:00 EDT 2008	69307	38.616835	-121.439146
4	6001 MCMAHON DR	33	95824	CA	2	1	797	2	Wed May 21 00:00:00 EDT 2008	81900	38.519470	-121.435768
...	...	...	...	...	...	...	...	...	...	...	...	...
980	9169 GARLINGTON CT	33	95829	CA	4	3	2280	2	Thu May 15 00:00:00 EDT 2008	232425	38.457679	-121.359620
981	6932 RUSKUT WAY	33	95823	CA	3	2	1477	2	Thu May 15 00:00:00 EDT 2008	234000	38.499893	-121.458890
982	7933 DAFFODIL WAY	4	95610	CA	3	2	1216	2	Thu May 15 00:00:00 EDT 2008	235000	38.708824	-121.256803
983	8304 RED FOX WAY	9	95758	CA	4	2	1685	2	Thu May 15 00:00:00 EDT 2008	235301	38.417000	-121.397424
984	3882 YELLOWSTONE LN	8	95762	CA	3	2	1362	2	Thu May 15 00:00:00 EDT 2008	235738	38.655245	-121.075915

6. Making the visualization for knowing how many each type house on the table

```
In [10]: plot = plt.figure(figsize=[10,10])
         sns.set_theme()
         sns.catplot(data = df_data_2, x = 'type', kind = 'count')
```

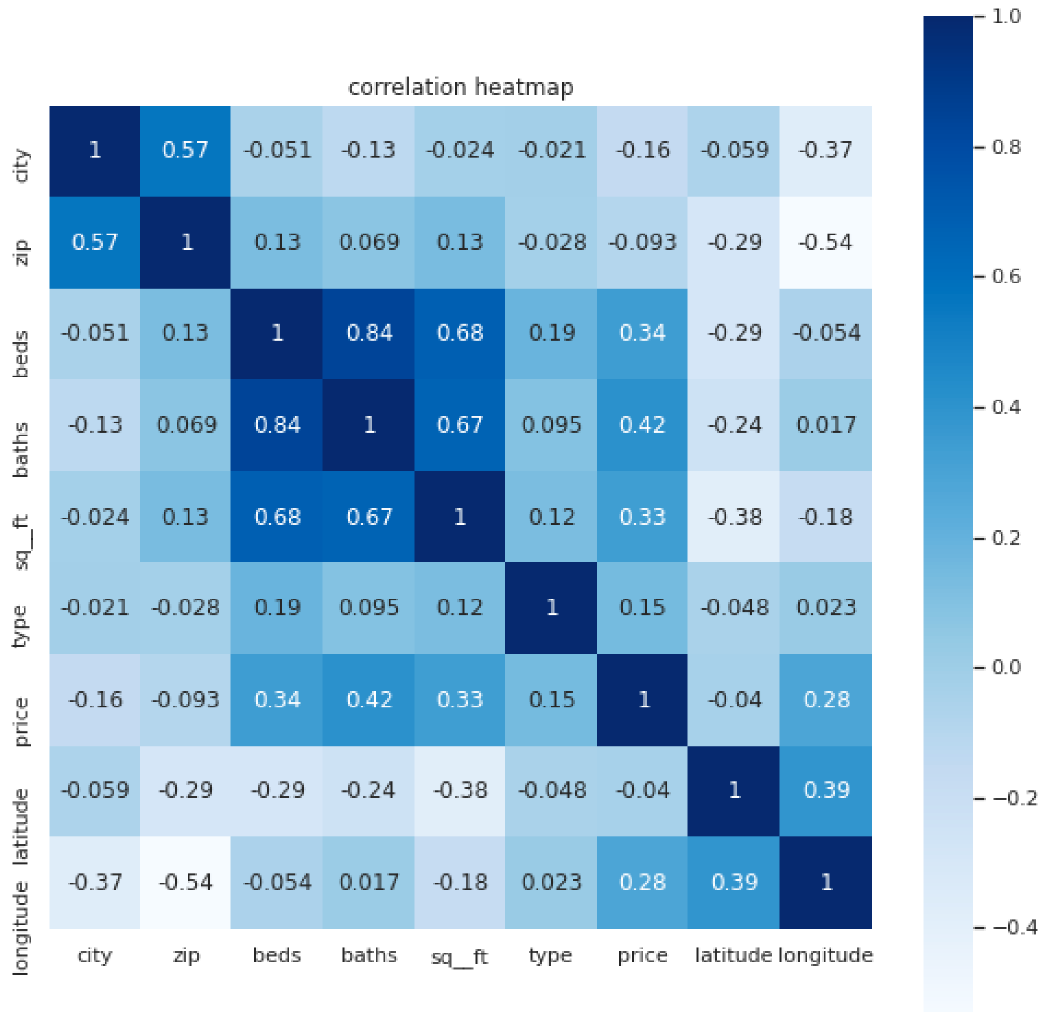
```
Out[10]: <seaborn.axisgrid.FacetGrid at 0x7f74984144c0>

         <Figure size 720x720 with 0 Axes>
```



7.
8. Getting the correlation number from table, so we can see which column has the closest relation with another column

```
In [12]: plot = plt.figure(figsize = [10,10])
         plt.title('correlation heatmap')
         corr = df_data_2.corr()
         sns.heatmap(corr, cbar = True, square = True, annot = True, cmap = 'Blues')
```

```
Out[12]: <AxesSubplot:title={'center':'correlation heatmap'}>
```



9. Set x and y

```
In [13]: #making x and y dataset
         X = df_data_2.drop(['street','zip','state','type','latitude','longitude','sale_date'], axis = 1)
```

```
In [14]: y = df_data_2['type']
```

10. Making train and test data

```
In [15]: x_train, x_test, y_train, y_test = train_test_split(X, y, test_size = 0.1, random_state = 10)
```

11. Call rfc

```
In [16]: rfc = RandomForestClassifier()
```

12. Train rfc

```
In [17]: rfc.fit(x_train, y_train)
```

```
Out[17]: RandomForestClassifier()
```

13. Predict all test set

```
In [18]: print('its the prediction result from x_test', rfc.predict(x_test))
```

```
its the prediction result from x_test [2 2 2 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 2 2 2 2 2 2 2 2 2 2]
```

14. Print (accuracy)

```
print('the accuracy score is',accuracy_score(y_test, rfc.predict(x_test)))
```

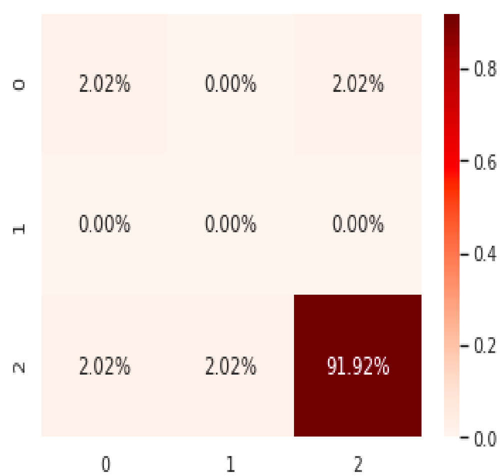
```
the accuracy score is 0.9393939393939394
```

15. Making confusion matrix accuracy

```
In [26]: cf_matrix = confusion_matrix(rfc.predict(x_test), y_test)
```

```
In [27]: import numpy as np
         sns.heatmap(cf_matrix/np.sum(cf_matrix), annot=True,
                     fmt='.2%', cmap='Reds')
```

```
Out[27]: <AxesSubplot:>
```



16. done